Optimisation of Quantum Hamiltonian Evolution

From two projection operators to local Hamiltonians

Apoorva Patel

Centre for High Energy Physics, Indian Institute of Science, Bangalore

23 June 2016, IQC, Waterloo, Canada

A. Patel, PoS(Lattice2014)324

A. Patel and A. Priyadarsini, arXiv:1503.01755

Abstract

Given a quantum Hamiltonian and its evolution time, the corresponding unitary evolution operator can be constructed in many different ways, corresponding to different trajectories between the desired end-points and different series expansions. A choice among these possibilities can then be made to obtain the best computational complexity and control over errors. It is shown how a construction based on Grover's algorithm scales linearly in time and logarithmically in the error bound, and is exponentially superior in error complexity to the scheme based on the straightforward application of the Lie-Trotter formula. The strategy is then extended first to simulation of any Hamiltonian that is a linear combination of two projection operators, and then to any local efficiently computable Hamiltonian. The key feature is to construct an evolution in terms of the largest possible steps instead of taking small time steps. Reflection operations and Chebyshev expansions are used to efficiently control the total error on the overall evolution, without worrying about discretisation errors for individual steps. We also use a digital implementation of quantum states that makes linear algebra operations rather simple to perform.

Motivation

Richard Feynman: Quantum computers are efficient simulators of quantum physical systems and models.

Classical simulations of quantum systems and models are not efficient.

Quantum superposition can sum multiple evolutionary paths contributing to a quantum process in one go, while classical simulators evaluate them one by one (as Markov chains).

Motivation

Richard Feynman: Quantum computers are efficient simulators of quantum physical systems and models.

Classical simulations of quantum systems and models are not efficient.

Quantum superposition can sum multiple evolutionary paths contributing to a quantum process in one go, while classical simulators evaluate them one by one (as Markov chains).

This advantage needs to be formalised in terms of computational complexity, for physical Hamiltonians.

Feynman (1982), Lloyd (1996), Aharanov and Ta-Shma (2003), Berry et al. (2007, 2013)

Motivation

Richard Feynman: Quantum computers are efficient simulators of quantum physical systems and models.

Classical simulations of quantum systems and models are not efficient.

Quantum superposition can sum multiple evolutionary paths contributing to a quantum process in one go, while classical simulators evaluate them one by one (as Markov chains).

This advantage needs to be formalised in terms of computational complexity, for physical Hamiltonians.

Feynman (1982), Lloyd (1996), Aharanov and Ta-Shma (2003), Berry et al. (2007, 2013)

Computational complexity of a problem is a measure of the physical resources required to solve it.

Space Time Energy

Conventional software considerations do not explicitly include energy.

Tradeoffs between resources are possible depending on their availability, e.g. space vs. time in parallel computers.

Computational Complexity

Existence of the universal Turing machine allows classification of computational complexity without reference to hardware.

Computational Complexity

Existence of the universal Turing machine allows classification of computational complexity without reference to hardware.

Computational complexity of a decision problem is specified in terms of the size of its input (output size is only one bit).

Problems with different output structures are reformulated as a sequence of decision problems, with successive verifiable bounds on the outputs (e.g. as in binary search).

For a specified tolerance level ϵ, the corresponding output size is $\log \epsilon$.

The complexity of the original problem is then the sum of complexities of the individual decision problems.

Computational Complexity

Existence of the universal Turing machine allows classification of computational complexity without reference to hardware.

Computational complexity of a decision problem is specified in terms of the size of its input (output size is only one bit).

Problems with different output structures are reformulated as a sequence of decision problems, with successive verifiable bounds on the outputs (e.g. as in binary search).

For a specified tolerance level ϵ, the corresponding output size is $\log \epsilon$.

The complexity of the original problem is then the sum of complexities of the individual decision problems.

It is therefore appropriate to specify the complexity of a general problem in terms of its input and output sizes.

This is a natural criterion for reversible computation. It is also suitable for extending finite precision analog computation to arbitrary precision digital computation.

Algorithmic Efficiency

A computational algorithm is efficient when the required resources are polynomial in its input and output sizes. These belong to the class P.

Nondeterministic algorithms requiring polynomial resources for verifying the answer belong to the class NP.

Many other classes have been defined: BPP, PSPACE, EXP, . . .

Algorithmic Efficiency

A computational algorithm is efficient when the required resources are polynomial in its input and output sizes. These belong to the class P.

Nondeterministic algorithms requiring polynomial resources for verifying the answer belong to the class NP.

Many other classes have been defined: BPP, PSPACE, EXP, . . .

Not enough attention has been paid to the output precision in the P vs. NP analysis. The task is relegated to analysis of numerical convergence of algorithms.

Popular importance sampling methods are not efficient.

Algorithmic Efficiency

A computational algorithm is efficient when the required resources are polynomial in its input and output sizes. These belong to the class P.

Nondeterministic algorithms requiring polynomial resources for verifying the answer belong to the class NP.

Many other classes have been defined: BPP, PSPACE, EXP, . . .

Not enough attention has been paid to the output precision in the P vs. NP analysis. The task is relegated to analysis of numerical convergence of algorithms.

Popular importance sampling methods are not efficient.

Consider the problem of finding the zero of a function.

Multiple trials: Central limit theorem gives $\epsilon = \epsilon_0/\sqrt{N_{\rm iter}} \Rightarrow N_{\rm iter} \sim \epsilon^{-2}$.

Bisection: Narrowing of the interval gives $\epsilon = \epsilon_0/2^{N_{\rm iter}} \Rightarrow N_{\rm iter} \sim \log(1/\epsilon)$.

Newton's method: Knowledge of the derivative gives $\epsilon = (\epsilon_0)^{2^{N_{\rm iter}}} \Rightarrow N_{\rm iter} \sim \log\log(1/\epsilon)$.

Efficient solution must exploit the structure in the problem.

Quantum Hamiltonian Simulation

Start from the initial quantum state $|\psi(0)\rangle$.

First evolve: $|\psi(T)\rangle = U(T)|\psi(0)\rangle, \quad U(T) = P[e^{-i\int_0^T H(t)dt}]$.

Then measure: $\langle O_a\rangle = \langle\psi(T)|O_a|\psi(T)\rangle$.

In typical problems, both these parts are executed probabilistically upto a specified tolerance level, say ϵ.

We address the first part: The problem is to determine the evolution operator $U(T)$, with accuracy $||\widetilde{U}(T) - U(T)|| < \epsilon$.

Efficient execution of the second part requires different techniques.

Quantum Hamiltonian Simulation

Start from the initial quantum state $|\psi(0)\rangle$.

First evolve: $|\psi(T)\rangle = U(T)|\psi(0)\rangle, \quad U(T) = P[e^{-i\int_0^T H(t)dt}]$.
Then measure: $\langle O_a \rangle = \langle \psi(T)|O_a|\psi(T)\rangle$.

In typical problems, both these parts are executed probabilistically upto a specified tolerance level, say ϵ.

We address the first part: The problem is to determine the evolution operator $U(T)$, with accuracy $||\widetilde{U}(T) - U(T)|| < \epsilon$.

Efficient execution of the second part requires different techniques.

In a finite N-dimensional Hilbert space, a generic $H(t)$ is a dense $N \times N$ matrix. That cannot be simulated efficiently.

Physical properties restrict the structure of $H(t)$, however.
Efficient simulations must exploit this Hamiltonian structure.

Useful Physical Features

Features commonly present in physical problems are:

(1) The Hilbert space is a tensor product of many small components (e.g. $N = 2^n$ for a system of qubits).

Useful Physical Features

Features commonly present in physical problems are:

(1) The Hilbert space is a tensor product of many small components (e.g. $N = 2^n$ for a system of qubits).

(2) The components have only local interactions (e.g. couplings with only a limited number of neighbours).

Such sparse Hamiltonians have $O(N)$ non-zero elements in the component basis.

"Local basis structure" can appear in different forms.

e.g. factorisable dense operator in FFT, multipole expansions for smooth long range interactions.

Useful Physical Features

Features commonly present in physical problems are:

(1) The Hilbert space is a tensor product of many small components (e.g. $N = 2^n$ for a system of qubits).

(2) The components have only local interactions (e.g. couplings with only a limited number of neighbours).

Such sparse Hamiltonians have $O(N)$ non-zero elements in the component basis.

"Local basis structure" can appear in different forms.

e.g. factorisable dense operator in FFT, multipole expansions for smooth long range interactions.

(3) The Hamiltonian is specified using a finite number of functions, with H_{ij} calculable from i, j (e.g. translationally invariant interactions).

Resources needed to just write down H should not influence the simulation complexity.

Useful Physical Features

Features commonly present in physical problems are:

(1) The Hilbert space is a tensor product of many small components (e.g. $N = 2^n$ for a system of qubits).

(2) The components have only local interactions (e.g. couplings with only a limited number of neighbours).

Such sparse Hamiltonians have $O(N)$ non-zero elements in the component basis.

"Local basis structure" can appear in different forms.

e.g. factorisable dense operator in FFT, multipole expansions for smooth long range interactions.

(3) The Hamiltonian is specified using a finite number of functions, with H_{ij} calculable from i, j (e.g. translationally invariant interactions).

Resources needed to just write down H should not influence the simulation complexity.

Such Hamiltonians can be mapped to graphs with bounded degree d (vertices $\leftrightarrow$ components, edges $\leftrightarrow$ interactions).

Above features permit SIMD simulations of these Hamiltonians with domain decomposition.

Useful Physical Features

Features commonly present in physical problems are:

(1) The Hilbert space is a tensor product of many small components (e.g. $N = 2^n$ for a system of qubits).

(2) The components have only local interactions (e.g. couplings with only a limited number of neighbours).

Such sparse Hamiltonians have $O(N)$ non-zero elements in the component basis.

"Local basis structure" can appear in different forms.

e.g. factorisable dense operator in FFT, multipole expansions for smooth long range interactions.

(3) The Hamiltonian is specified using a finite number of functions, with H_{ij} calculable from i, j (e.g. translationally invariant interactions).

Resources needed to just write down H should not influence the simulation complexity.

Such Hamiltonians can be mapped to graphs with bounded degree d (vertices $\leftrightarrow$ components, edges $\leftrightarrow$ interactions).

Above features permit SIMD simulations of these Hamiltonians with domain decomposition.

Efficient Hamiltonian simulation algorithms use resources that are polynomial in $\log(N)$, d, T and $\log(\epsilon)$. [Class P:P]

Lower Bounds on Complexity

Space: $d \log N$ (volume of system)
Time: T (duration of evolution)
Approximation errors: Should remain bounded
(i.e. independent of space and time) for best efficiency.

Lower Bounds on Complexity

Space: $d \log N$ (volume of system)
Time: T (duration of evolution)
Approximation errors: Should remain bounded
(i.e. independent of space and time) for best efficiency.

Control of Errors

In a digitised simulation, discretisation errors automatically appear.
There are many techniques to keep them under control,
which can be combined to achieve the best results:

(1) Equivalent evolutions with different step sizes.
Use effective theory with a large cutoff.

(2) Higher order expansions by superposing contributions.
k^{th} order expansion can give $\epsilon \sim 1/k!$ and $k \sim \log(1/\epsilon)/\log\log(1/\epsilon)$.
Control with superposed ancilla states can combine a large number of contributions.

(3) Selection of the answer using error correction codes.
Suppress the remnant error by powers, e.g. multiple runs and majority rule.

Evolution Strategy

Efficient simulation strategy has two major ingredients:

(A) Decompose the sparse Hamiltonian as a sum of non-commuting but block-diagonal parts, $H = \sum_{i=1}^{l} H_i$.

Blocks can be reduced in size to a mixture of 1×1 and 2×2 blocks.

Then each H_i can be easily and exactly exponentiated, with $\exp(-iH_i\tau)$ retaining the block-diagonal structure.

1×1 blocks exponentiate to phases. 2×2 blocks, $H^{(b)} = a_0 I + \vec{a} \cdot \vec{\sigma}$, have projection operator structure and can be interpreted as binary query oracles.

Evolution Strategy

Efficient simulation strategy has two major ingredients:

(A) Decompose the sparse Hamiltonian as a sum of non-commuting but block-diagonal parts, $H = \sum_{i=1}^{l} H_i$.

Blocks can be reduced in size to a mixture of 1×1 and 2×2 blocks.

Then each H_i can be easily and exactly exponentiated, with $\exp(-iH_i\tau)$ retaining the block-diagonal structure.

1×1 blocks exponentiate to phases. 2×2 blocks, $H^{(b)} = a_0 I + \vec{a} \cdot \vec{\sigma}$, have projection operator structure and can be interpreted as binary query oracles.

H_i can be identified by an edge-colouring algorithm for graphs, with distinct colours for overlapping edges.

Vizing: Any graph can be efficiently coloured with $d + 1$ colours.

Algorithms for bipartite graphs are simpler than the generic case. They need d colours.

Evolution Strategy

Efficient simulation strategy has two major ingredients:

(A) Decompose the sparse Hamiltonian as a sum of non-commuting but block-diagonal parts, $H = \sum_{i=1}^{l} H_i$.

Blocks can be reduced in size to a mixture of 1×1 and 2×2 blocks.

Then each H_i can be easily and exactly exponentiated, with $\exp(-iH_i\tau)$ retaining the block-diagonal structure.

1×1 blocks exponentiate to phases. 2×2 blocks, $H^{(b)} = a_0 I + \vec{a} \cdot \vec{\sigma}$, have projection operator structure and can be interpreted as binary query oracles.

H_i can be identified by an edge-colouring algorithm for graphs, with distinct colours for overlapping edges.

Vizing: Any graph can be efficiently coloured with $d + 1$ colours.

Algorithms for bipartite graphs are simpler than the generic case. They need d colours.

Identification of H_i provides a compressed labeling scheme to address individual blocks. The independent blocks can then be easily evolved in parallel (classically), or in superposition (quantum mechanically).

This strategy keeps the spatial scaling of the algorithm under control.

Example of Hamiltonian decomposition:

Discretised Laplacian in 1-dim can be decomposed as:

$$\begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & -1 & 2 & -1 & 0 & 0 & \cdots \\ \cdots & 0 & -1 & 2 & -1 & 0 & \cdots \\ \cdots & 0 & 0 & -1 & 2 & -1 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} =$$

$$\begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & -1 & 1 & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & -1 & 0 & \cdots \\ \cdots & 0 & 0 & -1 & 1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} + \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & 1 & -1 & 0 & 0 & \cdots \\ \cdots & 0 & -1 & 1 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & 1 & -1 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix}$$

This decomposition has the projection operator structure following from:

$$H = H_o + H_e, \;\; H_o^2 = 2H_o, \;\; H_e^2 = 2H_e.$$

Example of Hamiltonian decomposition:

Discretised Laplacian in 1-dim can be decomposed as:

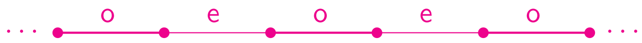
$$\begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & -1 & 2 & -1 & 0 & 0 & \cdots \\ \cdots & 0 & -1 & 2 & -1 & 0 & \cdots \\ \cdots & 0 & 0 & -1 & 2 & -1 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} =$$

$$\begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & -1 & 1 & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & -1 & 0 & \cdots \\ \cdots & 0 & 0 & -1 & 1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} + \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & 1 & -1 & 0 & 0 & \cdots \\ \cdots & 0 & -1 & 1 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & 1 & -1 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix}$$

This decomposition has the projection operator structure following from:

$$H = H_o + H_e,\ \ H_o^2 = 2H_o,\ \ H_e^2 = 2H_e.$$

Graphically, the bipartite break-up is:



H_o and H_e are identified by the last bit of the position label.

Eigenvalues of H are $4\sin^2(k/2)$. Those of H_o, H_e are $0, 2$.

Evolution Strategy (contd.)

(B) Use the discrete Lie-Trotter formula to exponentiate H, but with as large Δt as possible.

$$e^{-iHT} = e^{-i\sum_{i=1}^{l} H_i T} \approx \left(\prod_i e^{-iH_i\Delta t}\right)^n, \ \ n = T/\Delta t$$

This replacement maintains unitarity of the evolution,
but may not preserve other properties such as the energy.

Time-dependent Hamiltonians should be expanded about the mid-point of the interval Δt.

Evolution Strategy (contd.)

(B) Use the discrete Lie-Trotter formula to exponentiate H, but with as large Δt as possible.

$$e^{-iHT} = e^{-i\sum_{i=1}^{l} H_i T} \approx (\prod_i e^{-iH_i\Delta t})^n,\ \ n = T/\Delta t$$

This replacement maintains unitarity of the evolution,
but may not preserve other properties such as the energy.

Time-dependent Hamiltonians should be expanded about the mid-point of the interval Δt.

For block-diagonal H_i, exact exponentiation with large Δt keeps the temporal scaling of the algorithm under control.

When the exponent is proportional to a projection operator, the largest Δt makes the exponential a reflection operator.
$P = \frac{1}{2}(1 - \hat{n}\cdot\vec{\sigma}),\ P^2 = P\ \Rightarrow\ R = e^{\pm i\pi P} = 1 - 2P = \hat{n}\cdot\vec{\sigma},\ R^2 = I$

Evolution Strategy (contd.)

(B) Use the discrete Lie-Trotter formula to exponentiate H, but with as large Δt as possible.

$$e^{-iHT} = e^{-i\sum_{i=1}^{l} H_i T} \approx (\prod_i e^{-iH_i\Delta t})^n, \; n = T/\Delta t$$

This replacement maintains unitarity of the evolution,
but may not preserve other properties such as the energy.

Time-dependent Hamiltonians should be expanded about the mid-point of the interval Δt.

For block-diagonal H_i, exact exponentiation with large Δt keeps the temporal scaling of the algorithm under control.

When the exponent is proportional to a projection operator, the largest Δt makes the exponential a reflection operator.

$P = \frac{1}{2}(1 - \hat{n}\cdot\vec{\sigma}), \; P^2 = P \; \Rightarrow \; R = e^{\pm i\pi P} = 1 - 2P = \hat{n}\cdot\vec{\sigma}, \; R^2 = I$

The choice of large Δt also improves the error complexity from a power law dependence on ϵ to a logarithmic one.

That is not at all obvious, and needs to be demonstrated.

Summary of Results

Our simulation strategy yields the computational complexity

$$O\left(t\frac{\log(t/\epsilon)}{\log(\log(t/\epsilon))}\mathcal{C}\right)$$

where $\mathcal{C}$ is the simulation cost of a single time step (essentially the matrix-vector product $H|x\rangle$), which only weakly depends on t and ϵ.

Summary of Results

Our simulation strategy yields the computational complexity

$$O\left(t\,\frac{\log(t/\epsilon)}{\log(\log(t/\epsilon))}\mathcal{C}\right)$$

where $\mathcal{C}$ is the simulation cost of a single time step (essentially the matrix-vector product $H|x\rangle$), which only weakly depends on t and ϵ.

$\mathcal{C}$ characterises how computational complexity of classical implementation is improved in the quantum case, through conversion of independent parallel execution threads into quantum superposition.

Summary of Results

Our simulation strategy yields the computational complexity

$$O\left(t\frac{\log(t/\epsilon)}{\log(\log(t/\epsilon))}\mathcal{C}\right)$$

where $\mathcal{C}$ is the simulation cost of a single time step (essentially the matrix-vector product $H|x\rangle$), which only weakly depends on t and ϵ.

$\mathcal{C}$ characterises how computational complexity of classical implementation is improved in the quantum case, through conversion of independent parallel execution threads into quantum superposition.

For digital calculations with b-bit precision, discretisation errors are kept under control, with $b = \Omega(\log(t/\epsilon)\log(t/\epsilon))$.

Summary of Results

Our simulation strategy yields the computational complexity

$$O\left(t\,\frac{\log(t/\epsilon)}{\log(\log(t/\epsilon))}\mathcal{C}\right)$$

where $\mathcal{C}$ is the simulation cost of a single time step (essentially the matrix-vector product $H|x\rangle$), which only weakly depends on t and ϵ.

$\mathcal{C}$ characterises how computational complexity of classical implementation is improved in the quantum case, through conversion of independent parallel execution threads into quantum superposition.

For digital calculations with b-bit precision, discretisation errors are kept under control, with $b = \Omega(\log(t/\epsilon)\log(t/\epsilon)))$.

For l-sparse Hamiltonians, the cost $\mathcal{C}$ is $O(lNb^3)$ classically and $O(lnb^3)$ quantum mechanically.

Elements of H must be efficiently calculable functions of known parameters and the indices.

Illustration: Database Search

View database search as a Hamiltonian evolution problem.

The evolution is from the initial uniform superposition state $|s\rangle$ to a specific target state $|t\rangle$: $U(T)|s\rangle = |t\rangle$.

For a database of size N: $|\langle i|s\rangle| = 1/\sqrt{N}$, $\langle i|t\rangle = \delta_{it}$.

Illustration: Database Search

View database search as a Hamiltonian evolution problem.

The evolution is from the initial uniform superposition state $|s\rangle$ to a specific target state $|t\rangle$: $U(T)|s\rangle = |t\rangle$.

For a database of size N: $|\langle i|s\rangle| = 1/\sqrt{N},\ \langle i|t\rangle = \delta_{it}$.

Logic: Design a Hamiltonian to diffuse the wavefunction over the whole Hilbert space by kinetic energy (mean field version) and attract it towards the target by potential energy.

Illustration: Database Search

View database search as a Hamiltonian evolution problem.

The evolution is from the initial uniform superposition state $|s\rangle$ to a specific target state $|t\rangle$: $U(T)|s\rangle = |t\rangle$.

For a database of size N: $|\langle i|s\rangle| = 1/\sqrt{N},\ \langle i|t\rangle = \delta_{it}$.

Logic: Design a Hamiltonian to diffuse the wavefunction over the whole Hilbert space by kinetic energy (mean field version) and attract it towards the target by potential energy.

In simplest algorithms, the Hamiltonians depend only on $|s\rangle$ and $|t\rangle$. The unitary evolution is then a rotation in the 2-dim subspace formed by $|s\rangle$ and $|t\rangle$. Let

$$|t\rangle = \begin{pmatrix}1\\0\end{pmatrix},\ |t_\perp\rangle = \begin{pmatrix}0\\1\end{pmatrix},\ |s\rangle = \begin{pmatrix}1/\sqrt{N}\\ \sqrt{(N-1)/N}\end{pmatrix}.$$

A time-independent H rotates the state at a fixed rate:

$$|\psi\rangle \to U(t)|\psi\rangle,\ U(t) = \exp(-iHt) = \exp(-i\hat{n}_H \cdot \vec{\sigma}\omega t).$$

Farhi-Gutmann version

Continuous time evolution with $H_C = |s\rangle\langle s| + |t\rangle\langle t|$ gives:
$U(t) = \exp(-i\hat{n}\cdot\vec{\sigma}t/\sqrt{N}),\ \hat{n} = (\sqrt{(N-1)/N}, 0, 1/\sqrt{N})^T$.

The (unnormalised) eigenvectors of H are $|s\rangle \pm |t\rangle$.
The rotation axis $\hat{n}$ bisects the initial and target states.

Rotation by angle π on the Bloch sphere takes $|s\rangle$ to $|t\rangle$,
with evolution time $T = (\pi/2)\sqrt{N}$.

Grover version

Time evolution is discrete with the evolution operator
$U_G = -(I - 2|s\rangle\langle s|)(I - 2|t\rangle\langle t|) = (1 - \frac{2}{N})I + 2i\frac{\sqrt{N-1}}{N}\sigma_2$.

It is the discrete Lie-Trotter formula for H_s and H_t with $\Delta t_G = \pi$.
The rotation axis $\hat{n}_G = (0, 1, 0)^T$ is orthogonal to $\hat{n}$.

$U_G = \exp(-iH_G\tau)$ corresponds to the Hamiltonian
$H_G = i[|t\rangle\langle t|, |s\rangle\langle s|] = i(|t\rangle\langle s| - |s\rangle\langle t|)/\sqrt{N} = -\frac{\sqrt{N-1}}{N}\sigma_2$.

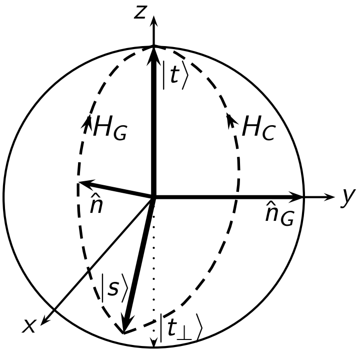
Non-trivial fact: H_G is the commutator of the two projection operators in H_C.

The evolution time step is: $\tau = \frac{2N}{\sqrt{N-1}} \sin^{-1} \frac{1}{\sqrt{N}}$.

Going from $|s\rangle$ to $|t\rangle$ requires Q steps along the geodesic:
$(U_G)^Q|s\rangle = |t\rangle, \; Q_T = \frac{1}{4}\cos^{-1}(\frac{2}{N} - 1)/\sin^{-1}(1/\sqrt{N}) \approx \frac{\pi}{4}\sqrt{N}.$

The evolution time step is: $\tau = \frac{2N}{\sqrt{N-1}} \sin^{-1} \frac{1}{\sqrt{N}}$.

Going from $|s\rangle$ to $|t\rangle$ requires Q steps along the geodesic:
$(U_G)^Q|s\rangle = |t\rangle,\ Q_T = \frac{1}{4}\cos^{-1}(\frac{2}{N} - 1)/\sin^{-1}(1/\sqrt{N}) \approx \frac{\pi}{4}\sqrt{N}$.



The two evolution trajectories are completely different ($\hat{n}$ and $\hat{n}_G$ are orthogonal). Only after a specific evolution time, corresponding to the solution of the search problem, they meet each other.

Adiabatic evolution follows the same trajectory as H_G.

Equivalent Evolutions

For database search: $U_C(T) = i(1 - 2|t\rangle\langle t|)(U_G)^{Q_T}$

For a more general evolution time $0 < t < T$, we have (analogous to the Euler angle decomposition):

$$U_C(t) = \exp(i\beta\sigma_3)\ (U_G)^{Q_t}\ \exp\left(i(\tfrac{\pi}{2} + \beta)\sigma_3\right),$$

$$Q_t = \frac{\sin^{-1}\left(\sqrt{\frac{N-1}{N}}\sin(t/\sqrt{N})\right)}{2\sin^{-1}(1/\sqrt{N})} \approx \frac{t}{2}\ ,\ \sigma_3 = 2|t\rangle\langle t| - 1,$$

$$\beta = -\frac{\pi}{4} - \frac{1}{2}\tan^{-1}\left(\frac{1}{\sqrt{N}}\tan(t/\sqrt{N})\right).$$

Equivalent Evolutions

For database search: $U_C(T) = i(1 - 2|t\rangle\langle t|)(U_G)^{Q_T}$

For a more general evolution time $0 < t < T$, we have (analogous to the Euler angle decomposition):

$$U_C(t) = \exp(i\beta\sigma_3)\ (U_G)^{Q_t}\ \exp\left(i(\tfrac{\pi}{2} + \beta)\sigma_3\right),$$

$$Q_t = \frac{\sin^{-1}\left(\sqrt{\frac{N-1}{N}}\sin(t/\sqrt{N})\right)}{2\sin^{-1}(1/\sqrt{N})} \approx \frac{t}{2}\ ,\ \sigma_3 = 2|t\rangle\langle t| - 1,$$

$$\beta = -\frac{\pi}{4} - \frac{1}{2}\tan^{-1}\left(\frac{1}{\sqrt{N}}\tan(t/\sqrt{N})\right).$$

Thus $U_C(t)$ can be expressed entirely in terms of projection operators, and the two evolutions are identical irrespective of the initial state and the evolution time.

H_G can be used to obtain the same evolution as H_C, even though they have completely different eigenvectors and eigenvalues.

Fractional oracle operator, $O_\phi = exp(i\phi|t\rangle\langle t|)$, is easily generated using an ancilla bit.

Complexity of Discretised Evolution

All continuous variables are discretised in digital computers.

That is needed for implementing fault-tolerant computation with control over bounded errors.

Discrete evolution step Δt has to be chosen so as to satisfy the overall error bound ϵ on the algorithm.

Complexity of Discretised Evolution

All continuous variables are discretised in digital computers.

That is needed for implementing fault-tolerant computation with control over bounded errors.

Discrete evolution step Δt has to be chosen so as to satisfy the overall error bound ϵ on the algorithm.

The simplest and the symmetric Lie-Trotter formulae are:

$$\begin{aligned}
e^{-i\sum_{i=1}^{l} H_i \Delta t} &= (e^{-iH_1\Delta t} ... e^{-iH_l\Delta t}) \times e^{-iE^{(2)}(\Delta t)^2} \\
e^{-i\sum_{i=1}^{l} H_i \Delta t} &= (e^{-iH_l\Delta t/2} ... e^{-iH_1\Delta t/2}) \\
&\times (e^{-iH_1\Delta t/2} ... e^{-iH_l\Delta t/2}) \times e^{-iE^{(3)}(\Delta t)^3}
\end{aligned}$$

with discretisation errors:

$$\begin{aligned}
E^{(2)} &= \frac{i}{24}\sum_{i<j}[H_i, H_j] + O(\Delta t) \\
E^{(3)} &= \frac{1}{24}\sum_{i<j}\{2[H_i,[H_i,H_j]] + [H_j,[H_i,H_j]]\} \\
&+ \frac{1}{12}\sum_{i<j<k}\{2[H_i,[H_j,H_k]] + [H_j,[H_i,H_k]]\} + O(\Delta t)
\end{aligned}$$

Small step size Δt:

For unitary operators, $||X|| = 1$. For n evolution steps, triangle and Cauchy-Schwarz inequalities bound the error:
$||X^m - Y^m|| = ||(X - Y)(X^{m-1} + \ldots + Y^{m-1})|| \leq m||X - Y||$.

So to keep the total discretisation error bounded, we need
$m||e^{-iE^{(k)}(\Delta t)^k} - I|| \approx m(\Delta t)^k ||E^{(k)}|| = m^{1-k} t^k ||E^{(k)}|| < \epsilon_1$.
R repetitions, and majority rule selection (not average)
of the result, further reduce the error to $2^{R-1}\epsilon_1^{\lceil R/2 \rceil} < \epsilon$.

Small step size Δt:

For unitary operators, $||X|| = 1$. For n evolution steps, triangle and Cauchy-Schwarz inequalities bound the error:
$||X^m - Y^m|| = ||(X-Y)(X^{m-1} + \ldots + Y^{m-1})|| \leq m||X-Y||$.

So to keep the total discretisation error bounded, we need
$m||e^{-iE^{(k)}(\Delta t)^k} - I|| \approx m(\Delta t)^k||E^{(k)}|| = m^{1-k}t^k||E^{(k)}|| < \epsilon_1$.
R repetitions, and majority rule selection (not average) of the result, further reduce the error to $2^{R-1}\epsilon_1^{\lceil R/2 \rceil} < \epsilon$.

With exact exponentiation of the individual H_i,
the computational cost $\mathcal{C}$ of a single step is independent of Δt.

The computational complexity of the whole evolution is then

$$O(mR\mathcal{C}) = O(t^{k/(k-1)}(||E^{(k)}||/\epsilon^{1/\lceil R/2 \rceil})^{1/(k-1)}R\mathcal{C}).$$

With power-law scaling in ϵ, this scheme is inefficient.

For the Hamiltonian H_C, $||E^{(2)}||$ and $||E^{(3)}||$ are $O(N^{-1/2})$.
For evolution time $T = \Theta(N^{1/2})$, its time complexity is linear.

Grover's discretisation:

Δt_G is chosen to make $\exp(-H_i \Delta t_G)$ reflection operators.

The large step size introduces an error because one may jump across the desired state instead of reaching it exactly.
In general, Q_t is not an integer, and has to be replaced by its nearest integer approximation $\lfloor Q_t + \frac{1}{2} \rfloor$.

Grover's discretisation:

Δt_G is chosen to make $\exp(-H_i \Delta t_G)$ reflection operators.

The large step size introduces an error because one may jump across the desired state instead of reaching it exactly.
In general, Q_t is not an integer, and has to be replaced by its nearest integer approximation $\lfloor Q_t + \frac{1}{2} \rfloor$.

The error probability for $U_C(t)$ is thus bounded by $1/N$, corresponding to half a rotation step. With R repetitions and majority rule selection of the result, the error probability becomes less than $2^{R-1}/N^{\lceil R/2 \rceil}$, which can be made less than any specified ϵ.

Grover's discretisation:

Δt_G is chosen to make $\exp(-H_i \Delta t_G)$ reflection operators.

The large step size introduces an error because one may jump across the desired state instead of reaching it exactly.
In general, Q_t is not an integer, and has to be replaced by its nearest integer approximation $\lfloor Q_t + \frac{1}{2} \rfloor$.

The error probability for $U_C(t)$ is thus bounded by $1/N$, corresponding to half a rotation step. With R repetitions and majority rule selection of the result, the error probability becomes less than $2^{R-1}/N^{\lceil R/2 \rceil}$, which can be made less than any specified ϵ.

The computational complexity of the total evolution is then

$$O(Q_t R \mathcal{C}_G) = O(\tfrac{t}{2}(-\tfrac{2\log\epsilon}{\log N})\mathcal{C}_G) = O(t\tfrac{\log(1/\epsilon)}{\log N}\mathcal{C}_G),$$

and the algorithm is efficient.

Key Observations

With a straightforward application of the Lie-Trotter formula, the algorithm has an error proportional to the number of steps m, and a power-law dependence of complexity on ϵ.

With the Lie-Trotter formula based on exact exponentiation of projection operators to reflection operators, the algorithm has an error independent of the evolution time, and a logarithmic dependence of complexity on ϵ.

Key Observations

With a straightforward application of the Lie-Trotter formula, the algorithm has an error proportional to the number of steps m, and a power-law dependence of complexity on ϵ.

With the Lie-Trotter formula based on exact exponentiation of projection operators to reflection operators, the algorithm has an error independent of the evolution time, and a logarithmic dependence of complexity on ϵ.

Algebraically, the Baker-Campbell-Hausdorff expansion simplifies for projection operators. That allows efficient implementation of the Lie-Trotter formula even for large Δt.

Key Observations

With a straightforward application of the Lie-Trotter formula, the algorithm has an error proportional to the number of steps m, and a power-law dependence of complexity on ϵ.

With the Lie-Trotter formula based on exact exponentiation of projection operators to reflection operators, the algorithm has an error independent of the evolution time, and a logarithmic dependence of complexity on ϵ.

Algebraically, the Baker-Campbell-Hausdorff expansion simplifies for projection operators. That allows efficient implementation of the Lie-Trotter formula even for large Δt.

With compressed labeling, operations on specific blocks are easily implemented as controlled unitary operations.

Euler angle decomposition allows easy conversion of rotations about arbitrary axes to rotations about fixed axes.

Digital Quantum State Implementation

A quantum state in an N-dimensional Hilbert space is:

$$|x\rangle = \sum_{j=0}^{N-1} x_j |j\rangle, \;\; \sum_{j=0}^{N-1} |x_j|^2 = 1.$$

This standard analog representation is not suitable for high precision calculations. So we use the digital representation instead, specified by the map:

$$|x\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle |x_j\rangle_b \; .$$

Digital Quantum State Implementation

A quantum state in an N-dimensional Hilbert space is:

$$|x\rangle = \sum_{j=0}^{N-1} x_j |j\rangle, \quad \sum_{j=0}^{N-1} |x_j|^2 = 1.$$

This standard analog representation is not suitable for high precision calculations. So we use the digital representation instead, specified by the map:

$$|x\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle |x_j\rangle_b \ .$$

This is a quantum state in a $(2^b N)$-dimensional Hilbert space, where $|x_j\rangle_b$ are basis vectors of a b-bit register representing the truncated value of x_j. It is fully entangled between the index state $|j\rangle$ and the register value state $|x_j\rangle_b$, with a unique $|x_j\rangle_b$ (out of 2^b possibilities) for every $|j\rangle$.

Digital Quantum State Implementation

A quantum state in an N-dimensional Hilbert space is:

$$|x\rangle = \sum_{j=0}^{N-1} x_j |j\rangle, \quad \sum_{j=0}^{N-1} |x_j|^2 = 1.$$

This standard analog representation is not suitable for high precision calculations. So we use the digital representation instead, specified by the map:

$$|x\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle |x_j\rangle_b \ .$$

This is a quantum state in a $(2^b N)$-dimensional Hilbert space, where $|x_j\rangle_b$ are basis vectors of a b-bit register representing the truncated value of x_j.
It is fully entangled between the index state $|j\rangle$ and the register value state $|x_j\rangle_b$, with a unique $|x_j\rangle_b$ (out of 2^b possibilities) for every $|j\rangle$.

No unitarity constraint is needed on the register values x_j.
A single index j controls the whole entangled state.

Index j can be handled in parallel (classically) or in superposition (quantum mechanically).

Freedom from the unitarity constraint allows simple implmentation of linear algebra operations, using only C-not and Toffoli gates of classical reversible logic, with the index j acting as control:

$$c|x\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle |cx_j\rangle_b \ ,$$
$$|x\rangle + |y\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle |x_j + y_j\rangle_b \ .$$

Freedom from the unitarity constraint allows simple implmentation of linear algebra operations, using only C-not and Toffoli gates of classical reversible logic, with the index j acting as control:

$$c|x\rangle \longrightarrow \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle|cx_j\rangle_b \ ,$$
$$|x\rangle + |y\rangle \longrightarrow \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle|x_j + y_j\rangle_b \ .$$

This freedom is useful because quantum states are never physically observed. Only expectation values of operators are observed.
So the digital representation is completed with the operator map:

$$O_a \to O_a \otimes O_b \quad \text{such that} \quad {}_b\langle x_j|O_b|x_l\rangle_b = N x_j^* x_l.$$

Freedom from the unitarity constraint allows simple implmentation of linear algebra operations, using only C-not and Toffoli gates of classical reversible logic, with the index j acting as control:

$$c|x\rangle \longrightarrow \tfrac{1}{\sqrt{N}} \textstyle\sum_{j=0}^{N-1} |j\rangle|cx_j\rangle_b \ ,$$
$$|x\rangle + |y\rangle \longrightarrow \tfrac{1}{\sqrt{N}} \textstyle\sum_{j=0}^{N-1} |j\rangle|x_j + y_j\rangle_b \ .$$

This freedom is useful because quantum states are never physically observed. Only expectation values of operators are observed.
So the digital representation is completed with the operator map:

$$O_a \to O_a \otimes O_b \quad \text{such that} \quad {}_b\langle x_j|O_b|x_l\rangle_b = N x_j^* x_l.$$

The solution, in a bit-wise fully factorised form, is

$$O_b = N V^\dagger (1+\sigma_1)^{\otimes b} V, \quad (1+\sigma_1)^{\otimes b} = 2^b |s\rangle_b\ {}_b\langle s|,$$

with the place-value operator, $V|x_j\rangle = x_j|x_j\rangle$, given by

$$V = \textstyle\sum_{k=0}^{b-1} 2^{-k} I^{\otimes k} \otimes \left(\frac{1-\sigma_3}{2}\right) \otimes I^{\otimes(b-k-1)}.$$

Computational complexity of expectation value calculations in the digital representation is $O(b^2)$ times that in the analog case.

Salient features:

• The same vector space coordinates x_j are used with two different metrics. Linear algebra is carried out with the Cartesian metric, while expectation values are evaluated with the metric O_b. The computation is deterministic throughout.

Salient features:

• The same vector space coordinates x_j are used with two different metrics. Linear algebra is carried out with the Cartesian metric, while expectation values are evaluated with the metric O_b. The computation is deterministic throughout.

• Non-unitary operations are mapped to unitary ones. Absence of unitarity constraint bypasses complicated amplitude amplification requirements.

Salient features:

• The same vector space coordinates x_j are used with two different metrics. Linear algebra is carried out with the Cartesian metric, while expectation values are evaluated with the metric O_b. The computation is deterministic throughout.

• Non-unitary operations are mapped to unitary ones. Absence of unitarity constraint bypasses complicated amplitude amplification requirements.

• The initial state can be created as

$$|0\rangle|0\rangle_b \xrightarrow{H^{\otimes n}\otimes I} \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle|0\rangle_b \xrightarrow{C_x} \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle|x_j(0)\rangle_b \ .$$

The final state observables are assumed to be efficiently computable from $x_j(T)$.

Salient features:

• The same vector space coordinates x_j are used with two different metrics. Linear algebra is carried out with the Cartesian metric, while expectation values are evaluated with the metric O_b. The computation is deterministic throughout.

• Non-unitary operations are mapped to unitary ones. Absence of unitarity constraint bypasses complicated amplitude amplification requirements.

• The initial state can be created as

$$|0\rangle|0\rangle_b \xrightarrow{H^{\otimes n}\otimes I} \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle|0\rangle_b \xrightarrow{C_x} \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle|x_j(0)\rangle_b \ .$$

The final state observables are assumed to be efficiently computable from $x_j(T)$.

• A measurement eigenstate can be identified by successive binary search. Binary Stern-Gerlach measurement can be performed as ($\delta_{jk} = j - k + 1$):

$$\frac{1}{\sqrt{2}}\sum_{j=0}^{1}|j\rangle|\delta_{jk}\rangle_1 \xrightarrow{(Cnot)_{21}} |1-k\rangle\frac{1}{\sqrt{2}}\sum_{j=0}^{1}|j\rangle_1 \xrightarrow{(Measure)_1} k \ .$$

Salient features:

• The same vector space coordinates x_j are used with two different metrics. Linear algebra is carried out with the Cartesian metric, while expectation values are evaluated with the metric O_b. The computation is deterministic throughout.

• Non-unitary operations are mapped to unitary ones. Absence of unitarity constraint bypasses complicated amplitude amplification requirements.

• The initial state can be created as

$$|0\rangle|0\rangle_b \xrightarrow{H^{\otimes n}\otimes I} \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle|0\rangle_b \xrightarrow{C_x} \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle|x_j(0)\rangle_b \ .$$

The final state observables are assumed to be efficiently computable from $x_j(T)$.

• A measurement eigenstate can be identified by successive binary search. Binary Stern-Gerlach measurement can be performed as $(\delta_{jk} = j - k + 1)$:

$$\frac{1}{\sqrt{2}}\sum_{j=0}^{1}|j\rangle|\delta_{jk}\rangle_1 \xrightarrow{(Cnot)_{21}} |1-k\rangle\frac{1}{\sqrt{2}}\sum_{j=0}^{1}|j\rangle_1 \xrightarrow{(Measure)_1} k \ .$$

• Linear operations on density matrices can also be carried out in a similar fashion, using the identity: $2^b\langle s|V|\rho_{ij}\rangle_{2b} = \rho_{ij}$.

Truncation Error

A digital computer with finite number of bits produces truncation errors. With b bits, the precision is $\delta = 2^{-b}$.

Addition, multiplication and polynomial evaluations respectively require $O(b)$, $O(b^2)$ and $O(b^3)$ resources.

Overflow/underflow limits the degree of the polynomial to be at most b.

With all functions approximated by accurate polynomials, and fixed axes rotations, b-bit precision needs $O(b^3)$ effort.

Truncation Error

A digital computer with finite number of bits produces truncation errors. With b bits, the precision is $\delta = 2^{-b}$.

Addition, multiplication and polynomial evaluations respectively require $O(b)$, $O(b^2)$ and $O(b^3)$ resources.

Overflow/underflow limits the degree of the polynomial to be at most b.

With all functions approximated by accurate polynomials, and fixed axes rotations, b-bit precision needs $O(b^3)$ effort.

Number of blocks is $O(N)$. Number of exponentiations of H_i, $N_{\exp}$, needed for the Lie-Trotter formula is $m(k-1)l$, which reduces to $2Q_t \approx t$ for the Grover version.

The truncation error can be always made negligible compared to the discretisation error, with the choice $N_{\exp}\delta = O(\epsilon)$, i.e. $b = \Theta(\log(m/\epsilon))$ or $\Theta(\log(t/\epsilon))$.

Truncation Error

A digital computer with finite number of bits produces truncation errors. With b bits, the precision is $\delta = 2^{-b}$.

Addition, multiplication and polynomial evaluations respectively require $O(b)$, $O(b^2)$ and $O(b^3)$ resources.

Overflow/underflow limits the degree of the polynomial to be at most b.

With all functions approximated by accurate polynomials, and fixed axes rotations, b-bit precision needs $O(b^3)$ effort.

Number of blocks is $O(N)$. Number of exponentiations of H_i, $N_{\rm exp}$, needed for the Lie-Trotter formula is $m(k-1)l$, which reduces to $2Q_t \approx t$ for the Grover version.

The truncation error can be always made negligible compared to the discretisation error, with the choice $N_{\rm exp}\delta = O(\epsilon)$, i.e. $b = \Theta(\log(m/\epsilon))$ or $\Theta(\log(t/\epsilon))$.

The cost of a single step then scales as $\mathcal{C} = O(\log N(\log(t/\epsilon))^3)$, and the algorithm is efficient.

Extensions

For more general Hamiltonians, the evolution strategy can be chosen as a rapidly converging series expansion. Reflection operators have the largest spectral gap among unitary operators, and lead to fast convergence.

Tal-Ezer and Kosloff (1984), Berry et al. (2015)

Extensions

For more general Hamiltonians, the evolution strategy can be chosen as a rapidly converging series expansion. Reflection operators have the largest spectral gap among unitary operators, and lead to fast convergence.

Tal-Ezer and Kosloff (1984), Berry et al. (2015)

Two Projection Operators:

Let $H = H_1 + H_2$, $H_i^2 = H_i$, $R_i = I - 2H_i$, $r_0(0) = 1$, $r_k(0) = 0$. Then

$$\begin{aligned} e^{it}\exp(-iHt) &= \exp\left(i(R_1 + R_2)\frac{t}{2}\right) \\ &= r_0(t)\ I + \sum_{k=1}^{\infty} r_k(t)\left[(R_1 R_2 R_1 \ldots)_k + (R_2 R_1 R_2 \ldots)_k\right]. \end{aligned}$$

Truncated series can be efficiently evaluated as nested products.

Extensions

For more general Hamiltonians, the evolution strategy can be chosen as a rapidly converging series expansion. Reflection operators have the largest spectral gap among unitary operators, and lead to fast convergence.

Tal-Ezer and Kosloff (1984), Berry et al. (2015)

Two Projection Operators:

Let $H = H_1 + H_2$, $H_i^2 = H_i$, $R_i = I - 2H_i$, $r_0(0) = 1$, $r_k(0) = 0$. Then

$$\begin{aligned} e^{it} \exp(-iHt) &= \exp\left(i(R_1 + R_2)\frac{t}{2}\right) \\ &= r_0(t)\ I + \sum_{k=1}^{\infty} r_k(t)\left[(R_1 R_2 R_1 \ldots)_k + (R_2 R_1 R_2 \ldots)_k\right]. \end{aligned}$$

Truncated series can be efficiently evaluated as nested products.

Differentiation of this expansion provides recurrence relations for the coefficients ($k \geq 1$):

$$\frac{dr_0(t)}{dt} = ir_1(t)\ , \quad \frac{dr_k(t)}{dt} = \frac{i}{2}\left(r_{k+1}(t) + r_{k-1}(t)\right)\ .$$

The solutions are the Bessel functions:

$$r_0(t) = J_0(t)\ , \quad r_k(t) = i^k J_k(t)\ , \quad |r_k(t)| = O(t^k/(2^k k!))\ .$$

The solutions are the Bessel functions:

$$r_0(t) = J_0(t) \ , \quad r_k(t) = i^k J_k(t) \ , \quad |r_k(t)| = O(t^k/(2^k k!)) \ .$$

Summation of the series to order p requires $2p$ executions of the linear algebra operation $(rI + R)|x\rangle$.

Multiplication of $|x\rangle$ by block-diagonal R_i can be performed by shuffling elements of $|x\rangle$ within each block.

The solutions are the Bessel functions:

$$r_0(t) = J_0(t)\ ,\quad r_k(t) = i^k J_k(t)\ ,\quad |r_k(t)| = O(t^k/(2^k k!))\ .$$

Summation of the series to order p requires $2p$ executions of the linear algebra operation $(rI + R)|x\rangle$.

Multiplication of $|x\rangle$ by block-diagonal R_i can be performed by shuffling elements of $|x\rangle$ within each block.

With $t = m\Delta t$ and $||R_i|| = 1$, the truncation error is:

$$2m\frac{(\Delta t)^{p+1}}{2^{p+1}(p+1)!}\left(1 - \frac{\Delta t}{2(p+2)}\right)^{-1} < \epsilon\ .$$

When $\Delta t = \Theta(1)$, formally $p = O(\log(t/\epsilon)/\log(\log(t/\epsilon)))$.

Numerical tests indicate that $\Delta t = \pi$ is a good choice.

The solutions are the Bessel functions:

$$r_0(t) = J_0(t)\ ,\quad r_k(t) = i^k J_k(t)\ ,\quad |r_k(t)| = O(t^k/(2^k k!))\ .$$

Summation of the series to order p requires $2p$ executions of the linear algebra operation $(rI + R)|x\rangle$.

Multiplication of $|x\rangle$ by block-diagonal R_i can be performed by shuffling elements of $|x\rangle$ within each block.

With $t = m\Delta t$ and $||R_i|| = 1$, the truncation error is:

$$2m \frac{(\Delta t)^{p+1}}{2^{p+1}(p+1)!} \left(1 - \frac{\Delta t}{2(p+2)}\right)^{-1} < \epsilon\ .$$

When $\Delta t = \Theta(1)$, formally $p = O(\log(t/\epsilon)/\log(\log(t/\epsilon)))$.

Numerical tests indicate that $\Delta t = \pi$ is a good choice.

The computational complexity is then, with $\mathcal{C} = O(nb^3)$,

$$O(2mp\mathcal{C}) = O\left(t \frac{\log(t/\epsilon)}{\log(\log(t/\epsilon))} \mathcal{C}\right)\ .$$

Local Bounded Hamiltonians:

Chebyshev expansion provides a uniform approximation to any bounded function. Scale the Hamiltonian such that its spectrum is within the domain $[-1,1]$ of the Chebyshev polynomials $T_n(x) = \cos(n \cos^{-1} x)$.

$e^{-iHt} = \sum_{k=0}^{\infty} C_k(t)\ T_k(H)$ has the expansion coefficients:

$$C_0 = \frac{1}{\pi} \int_0^\pi e^{-it\cos\theta} d\theta = J_0(t) \ ,$$
$$C_{k>0} = \frac{2}{\pi} \int_0^\pi e^{-it\cos\theta} \cos(k\theta)\ d\theta = 2(-i)^k J_k(t) \ .$$

Local Bounded Hamiltonians:

Chebyshev expansion provides a uniform approximation to any bounded function. Scale the Hamiltonian such that its spectrum is within the domain $[-1,1]$ of the Chebyshev polynomials $T_n(x) = \cos(n\cos^{-1}x)$.

$e^{-iHt} = \sum_{k=0}^{\infty} C_k(t)\ T_k(H)$ has the expansion coefficients:

$$C_0 = \frac{1}{\pi}\int_0^{\pi} e^{-it\cos\theta} d\theta = J_0(t)\ ,$$
$$C_{k>0} = \frac{2}{\pi}\int_0^{\pi} e^{-it\cos\theta}\cos(k\theta)\ d\theta = 2(-i)^k J_k(t)\ .$$

Partially summed reflection operator series is actually the Chebyshev expansion (matrix functions are defined as their power series expansions):

$$T_k\left(-\frac{R_1+R_2}{2}\right) = \frac{(-1)^k}{2}\left((R_1R_2\ldots)_k + (R_2R_1\ldots)_k\right).$$

Local Bounded Hamiltonians:

Chebyshev expansion provides a uniform approximation to any bounded function. Scale the Hamiltonian such that its spectrum is within the domain $[-1, 1]$ of the Chebyshev polynomials $T_n(x) = \cos(n \cos^{-1} x)$.

$e^{-iHt} = \sum_{k=0}^{\infty} C_k(t)\ T_k(H)$ has the expansion coefficients:

$$C_0 = \frac{1}{\pi} \int_0^{\pi} e^{-it\cos\theta} d\theta = J_0(t)\ ,$$
$$C_{k>0} = \frac{2}{\pi} \int_0^{\pi} e^{-it\cos\theta} \cos(k\theta)\ d\theta = 2(-i)^k J_k(t)\ .$$

Partially summed reflection operator series is actually the Chebyshev expansion (matrix functions are defined as their power series expansions):

$$T_k\left(-\frac{R_1+R_2}{2}\right) = \frac{(-1)^k}{2}\left((R_1 R_2 \ldots)_k + (R_2 R_1 \ldots)_k\right).$$

Clenshaw's algorithm, based on the recursion relation,

$$T_{k+1}(H) = 2H\ T_k(H) - T_{k-1}(H).$$

efficiently sums a truncated Chebyshev expansion for $e^{-iHt}|x\rangle$, with p sparse matrix-vector products. Note that $||T_k(H)|| \leq 1$.

The computational complexity is $O(mp\mathcal{C}_C)$, with $\mathcal{C}_C = O(nb^3)$.

An Alternate Strategy:

Chebyshev expansion coefficients $J_k(t)$ are bounded for any value of t, and rapidly fall off for $k > t$. So e^{-iHt} can be evaluated at one shot, without subdividing the time interval into multiple steps.

This requires H to be time independent, otherwise evolution has to be performed piece-wise over time intervals where H is effectively constant.

An Alternate Strategy:

Chebyshev expansion coefficients $J_k(t)$ are bounded for any value of t, and rapidly fall off for $k > t$. So e^{-iHt} can be evaluated at one shot, without subdividing the time interval into multiple steps.

This requires H to be time independent, otherwise evolution has to be performed piece-wise over time intervals where H is effectively constant.

Truncation of the expansion at order p then results in error

$$\sum_{k=p+1}^{\infty} |C_k(t)| \leq \frac{t^{p+1}}{2^p(p+1)!} \left(1 - \frac{t}{2(p+2)}\right)^{-1} < \epsilon \ .$$

Its control needs $p > et/2$, and the formal bound is $p = O(t\epsilon^{-1/t}) = O(t + \log(1/\epsilon))$.

It is assumed that subleading terms in Bessel function expansions are negligible.

An Alternate Strategy:

Chebyshev expansion coefficients $J_k(t)$ are bounded for any value of t, and rapidly fall off for $k > t$. So e^{-iHt} can be evaluated at one shot, without subdividing the time interval into multiple steps.

This requires H to be time independent, otherwise evolution has to be performed piece-wise over time intervals where H is effectively constant.

Truncation of the expansion at order p then results in error

$$\sum_{k=p+1}^{\infty} |C_k(t)| \leq \frac{t^{p+1}}{2^p(p+1)!} \left(1 - \frac{t}{2(p+2)}\right)^{-1} < \epsilon \ .$$

Its control needs $p > et/2$, and the formal bound is $p = O(t\epsilon^{-1/t}) = O(t + \log(1/\epsilon))$.

It is assumed that subleading terms in Bessel function expansions are negligible.

The computational complexity is $O(p\mathcal{C}_C)$, with $\mathcal{C}_C = O(nb^3)$.

An Alternate Strategy:

Chebyshev expansion coefficients $J_k(t)$ are bounded for any value of t, and rapidly fall off for $k > t$. So e^{-iHt} can be evaluated at one shot, without subdividing the time interval into multiple steps.

This requires H to be time independent, otherwise evolution has to be performed piece-wise over time intervals where H is effectively constant.

Truncation of the expansion at order p then results in error

$$\sum_{k=p+1}^{\infty} |C_k(t)| \le \frac{t^{p+1}}{2^p (p+1)!} \left(1 - \frac{t}{2(p+2)}\right)^{-1} < \epsilon \ .$$

Its control needs $p > et/2$, and the formal bound is $p = O(t\epsilon^{-1/t}) = O(t + \log(1/\epsilon))$.

It is assumed that subleading terms in Bessel function expansions are negligible.

The computational complexity is $O(p\mathcal{C}_C)$, with $\mathcal{C}_C = O(nb^3)$.

The computational effort to evaluate $J_k(t)$ upto order p, with $b = \Omega(\log(p/\epsilon))$ bit precision, is $\Theta(pb^2)$, and so does not alter the computational complexity. The procedure runs the recursion relation,

$$J_{k-1}(t) = \frac{2k}{t} J_k(t) - J_{k-1}(t) \ ,$$

in descending order, and normalises using $J_0(t) + 2\sum_{k=1}^{\infty} J_{2k}(t) = 1$.

Digital State Implementation

Efficient multiplication of the sparse Hamiltonian with a vector needs block decomposition, to convert the computational complexity from classical $O(N)$ to quantum $O(n)$.

Digital State Implementation

Efficient multiplication of the sparse Hamiltonian with a vector needs block decomposition, to convert the computational complexity from classical $O(N)$ to quantum $O(n)$.

Edge-colouring of the Hamiltonian graph with l colours gives l Hamiltonian parts, each containing $O(N/2)$ mutually independent 2×2 blocks.

Diagonal elements H_{jj} can be absorbed into the 2×2 blocks according to convenience.
A single edge represents the off-diagonal elements $H_{j,j+\mu} = H^*_{j+\mu,j}$.

Digital State Implementation

Efficient multiplication of the sparse Hamiltonian with a vector needs block decomposition, to convert the computational complexity from classical $O(N)$ to quantum $O(n)$.

Edge-colouring of the Hamiltonian graph with l colours gives l Hamiltonian parts, each containing $O(N/2)$ mutually independent 2×2 blocks.

Diagonal elements H_{jj} can be absorbed into the 2×2 blocks according to convenience.
A single edge represents the off-diagonal elements $H_{j,j+\mu} = H^*_{j+\mu,j}$.

The 2×2 block multiplication is straightforward, with off-diagonal multiplication carried out using the swap operation:

$$|j\rangle|y_j\rangle_b + |j+\mu\rangle|y_{j+\mu}\rangle_b \xrightarrow{S} |j\rangle|y_{j+\mu}\rangle_b + |j+\mu\rangle|y_j\rangle_b \ .$$

$S = \sigma_1 \otimes I^{\otimes b}$ acts on the subspace $\{|j\rangle, |j+\mu\rangle\} \otimes \{|y_j\rangle_b, |y_{j+\mu}\rangle_b\}$.

The only difference between classical and quantum algorithms is the exponential advantage of linear superposition of states.

Going Beyond

• Evolutions using a large step size can be looked upon as simulation of an effective Hamiltonian. The effective Hamiltonian can be quite different from the original Hamiltonian, and yet dramatically improve the accuracy of the simulation. Finding appropriate equivalent Hamiltonians can be useful in speeding up a variety of problems, e.g. adiabatic evolutions.

Going Beyond

- Evolutions using a large step size can be looked upon as simulation of an effective Hamiltonian. The effective Hamiltonian can be quite different from the original Hamiltonian, and yet dramatically improve the accuracy of the simulation. Finding appropriate equivalent Hamiltonians can be useful in speeding up a variety of problems, e.g. adiabatic evolutions.

- Our digital representation bypasses the unitary constraint, allowing a map between non-unitary linear algebra operations and unitary operators. This framework can help in construction of class P:P quantum algorithms for many linear algebra problems, e.g. solution of $Ax = b$.

Going Beyond

• Evolutions using a large step size can be looked upon as simulation of an effective Hamiltonian. The effective Hamiltonian can be quite different from the original Hamiltonian, and yet dramatically improve the accuracy of the simulation. Finding appropriate equivalent Hamiltonians can be useful in speeding up a variety of problems, e.g. adiabatic evolutions.

• Our digital representation bypasses the unitary constraint, allowing a map between non-unitary linear algebra operations and unitary operators. This framework can help in construction of class P:P quantum algorithms for many linear algebra problems, e.g. solution of $Ax = b$.

• Projection operator identities are implicit but essential in the simpflication of our algebra. In particular, the reduction

$$[P,[P,[P,X]]] = [P,X] ,$$

for a projection operator P, can simplify the Baker-Campbell-Hausdorff expansion in other applications as well.

References

R.P. Feynman, Simulating Physics with Computers,
Int. J. Theor. Phys. 21 (1982) 467-488

S. Lloyd, Universal Quantum Simulators,
Science 273 (1996) 1073-1078

D. Aharonov and A. Ta-Shma,
Adiabatic Quantum State Generation and Statistical Zero Knowledge,
Proc. 35th Annual ACM Symp. on Theory of Computing, ACM (2003) 20-29

D.W. Berry, G. Ahokas, R. Cleve and B.C. Sanders,
Efficient Quantum Algorithms for Simulating Sparse Hamiltonians,
Comm. Math. Phys. 270 (2007) 359-371

A.M. Childs and R. Kothari,
Simulating Sparse Hamiltonians with Star Decompositions,
Proc. TQC2010, Lecture Notes in Comp. Sci. 6519 (2011) 94-103

D.W. Berry, A.M. Childs, R. Cleve, R. Kothari and R.D. Somma,
Exponential Improvement in Precision for Simulating Sparse Hamiltonians,
Proc. 46th Annual ACM Symp. on Theory of Computing, ACM (2014) 283-292

A. Patel, Optimisation of Quantum Evolution Algorithms,
Proc. 32nd International Symp. on Lattice Field Theory, PoS(Lattice2014)324

H. De Raedt, Product Formula Algorithms for Solving the Time-Dependent Schrödinger
Equation, Comp. Phys. Rep. 7 (1987) 1-72

J.L. Richardson, Visualizing Quantum Scattering on the CM-2 Supercomputer,
Comp. Phys. Comm. 63 (1991) 84-94

L.K. Grover, From Schrödinger's Equation to the Quantum Search Algorithm,
Pramana 56 (2001) 333-348

E. Farhi and S. Gutmann, An Analog Analogue of a Digital Quantum Computation,
Phys. Rev. A 57 (1998) 2403-2406

L.K. Grover, A Fast Quantum Mechanical Algorithm for Database Search,
Proc. 28th Annual ACM Symp. on Theory of Computing, ACM (1996) 212-219

M.A. Nielsen and I.L. Chuang, Quantum Computation and Quantum Information,
Cambridge University Press (2000)

H. Tal-Ezer and R. Kosloff, An Accurate and Efficient Scheme for Propagating the
Time Dependent Schrödinger Equation, J. Chem. Phys. 81 (1984) 3967-3971

D.W. Berry, A.M. Childs, R. Cleve, R. Kothari and R.D. Somma,
Simulating Hamiltonian Dynamics with a Truncated Taylor Series,
Phys. Rev. Lett. 114 (2015) 090502

G.B. Arfken, H.J. Weber and F.E. Harris,
Mathematical Methods for Physicists: A Comprehensive Guide,
Seventh Edition, Academic Press (2011)

M. Abramowitz and I.A. Stegun (Eds.),
Handbook of Mathematical Functions: With Formulas, Graphs and Mathematical Tables,
Dover Publications (1965)

R. Achilles and A. Bonfiglioli,
The Early Proofs of the Theorem of Campbell, Baker, Hausdorff, and Dynkin,
Arch. Hist. Exact. Sci. 66 (2012) 295-358